IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:
Nimrod Megiddo — Atty. Docket No.: ARC920030085US1

Serial No.: 10/723,850 — Group Art Unit: 2191

Filed: November 25, 2003 — Examiner: Junchun Wu

For: A SYSTEM AND METHOD FOR AUTONOMIC OPTIMIZATION BY COMPUTER PROGRAMS

Commissioner of Patents
P.O. BOX 1450
Alexandria, VA 22313-1450

DECLARATION UNDER 37 C.F.R. §1.131

I, Nimrod Megiddo, the inventor of the invention defined by claims 1-25 of U.S. Patent Application Serial No. 10/723,850 hereby declare the following:

[0001] The purpose of this declaration is to prove that I conceived the claimed invention prior to the earliest effective prior art date of U.S. Patent Application Publication No. 2005/0100209A1 to Lewis et al. which is presently understood to be July 2, 2003. The following shows that I conceived my invention prior to July 2, 2003 and that I was diligent from my date of conception to its reduction to practice and was further diligent to the date of the filing of my patent application, which has a filing date of November 25, 2003 (hereinafter referred to as the "Patent Application"). I first disclosed the invention "A METHOD FOR AUTONOMIC OPTIMIZATION BY COMPUTER PROGRAMS" on a date before July 2, 2003, opened as Disclosure No. ARC820020057, which is dated before July 2, 2003.

[0002] I am the only inventor of the subject matter claimed in claims 1-25 of the Patent Application.

[0003] During all time periods mentioned herein, and specifically between my conception date and the filing date of the Patent Application, all activities described herein occurred in the United States.

[0004] Proof of the conception of the claimed invention prior to July 2, 2003, and diligence in reducing the invention to practice and filing the Patent Application is demonstrated in the attached Exhibit, labeled as Exhibit A.

[0005] As shown in Exhibit A, which is an invention disclosure form typically used by the designated Assignee, International Business Machines Corporation, I conceived the claimed invention at a date prior to July 2, 2003. As permitted by MPEP §715.07, the dates on Exhibit A have been removed; however, I hereby declare that all dates corresponding to the conception date and reduction to practice occurred prior to July 2, 2003. Further, the invention was actually conceived before Exhibit A was prepared. Therefore, our conception date actually predates Exhibit A.

[0006] Exhibit A specifically discloses the claimed invention as defined by the independent claims. For example, independent claim 1 recites, "A method of instructing a computer program to self-optimize, said method comprising: inputting a selection command that selects one function from a list of pre-selected functions for input into said computer program at a point of choice determined by a programmer, wherein each function from said list of pre-selected functions is associated with a reward; and allowing a learning protocol in said computer program to track and reward said one function that is selected and to determine an approximate optimal choice of operation of said computer program based on said selection command."

[0007] For example, independent claim 7 recites, "A method of optimizing a computer program, said method comprising: specifying at least one point of choice, determined by a programmer, in said computer program; defining a set of alternate choices at each point of choice, wherein said set of alternate choices include operational choices comprising: inputting a selection command that selects one function from a list of pre-selected functions for input into said computer program, wherein each function from said list of pre-selected functions is associated with a reward; allowing a learning protocol in said computer program to track and reward said one function that is selected to determine an approximate optimal operation of said computer program based on said selection command; and setting at least one feedback point for said each point of choice.

[0008] For example, independent claim 13 recites, "A program storage device readable by computer, tangibly embodying a program of instructions executable by said computer to perform a method of instructing a computer program to self-optimize, said method comprising: inputting a selection command that selects one function from a list of pre-selected functions for input into said computer program at a point of choice, determined by a programmer, wherein each function from said list of pre-selected functions is associated with a reward; and allowing a learning protocol in said computer program to track and reward said one function that is selected and to determine an approximate optimal choice of operation of said computer program based on at least said selection command.

[0009] For example, independent claim 19 recites, " A computer system comprising: a pre-complier that inputs a selection command at a point of choice, determined by a programmer, into a computer program that runs on a computer, said selection command selecting one function from a list of pre-selected functions for input into said computer program, wherein each function from said list of pre-selected functions is associated with a reward; and a processor adapted to execute a learning protocol in said computer program to track and reward said one function that is selected and determine an approximate optimal operation of said computer program based on at least said selection command.

[0010] Exhibit A describes the above features. In fact, the descriptions provided in Exhibit A served as the basis for the specification, drawings, and claims of the Patent Application. The features provided in dependent claims 5-6, 8, 11-12, 17-18, and 23-24 are generally inferred in Exhibit A.

[0011] I was diligent from the date of conception in reducing the invention to practice and in pursuing, preparing, and filing the Patent Application.

[0012] I am informed and believe that on September 2, 2003, a patent attorney was instructed to prepare a patent application that eventually became the Patent Application, and the Patent Application was eventually prepared and filed on November 25, 2003.

[0013] The foregoing declarations are made according to my best recollection upon review of the appropriate documents and notes, and I hereby acknowledge that willful false statements and the like are punishable by fine or imprisonment, or both (18 USC §1001) and may jeopardize the validity of the application or any patent issuing thereon. All statements made herein are made of my own knowledge and are true and all statements that are made on information and belief are believed to be true.

[illegible] June 2, 2008

Nimrod Megiddo Date

EXHIBIT A

Disclosure ARC8-2003-0069

Prepared for and/or by an IBM Attorney - IBM Confidential

Created By Nimrod Megiddo [redacted]
Last Modified By Leonard T Guzman [redacted]

Required fields are marked with the asterisk(*) and must be filled in to complete the form.

*Title of disclosure (in English)
A Method for Autonomic Optimization by Computer Programs
A Method for Autonomic Optimization by Computer Programs

Summary

Status		Under Evaluation
Final Deadline		
Final Deadline Reason		
*Processing Location		Almaden
*Functional Area	select	(K53) K53 - CS Principals & Methodologies - (D. Williamson)
Attorney/Patent Professional		Leonard T Guzman/Almaden/IBM
IDT Team	select	Leonard T Guzman/Almaden/IBM Ronald Fagin/Almaden/IBM David P Williamson/Almaden/IBM Iris Eiron/Almaden/IBM Daniel M Shiffman/Almaden/IBM Tushar Chandra/Watson/IBM
Submitted Date		[redacted]
*Owning Division	select	RES
Incentive Program		
Lab		
*Technology Code		610
PVT Score		35

Inventors with a Blue Pages entry

Inventors: Nimrod Megiddo/Almaden/IBM

	Inventor Name	Inventor Serial	Div/Dept	Inventor Phone	Manager Name
>	Megiddo, Nimrod	645231	22/K53D	457-1786	Fagin, Ronald

> denotes primary contact

Inventors without a Blue Pages entry

IDT Selection

Attorney/Patent Professional	Leonard T Guzman/Almaden/IBM
IDT Team	Leonard T Guzman/Almaden/IBM

Ronald Fagin/Almaden/IBM
David P Williamson/Almaden/IBM
Iris Eiron/Almaden/IBM
Daniel M Shiffman/Almaden/IBM
Tushar Chandra/Watson/IBM

Response Due to IP&L [redacted]

***Main Idea**

1. Background: What is the problem solved by your invention? Describe known solutions to this problem (if any). What are the drawbacks of such known solutions, or why is an additional solution required? Cite any relevant technical documents or references.

Software engineers often have various alternate ways of implementing algorithms and they fix choices within the code based on current beliefs about future environments. For example, an online scheduling algorithm picks tasks for execution, attempting to optimize the system according to some criterion, but the code designer may not know what the best choices are in various situations. Quite often, the code designer lacks the expertise that is required for making the good choices. In many situations, the good choices depend on the environment and it is quite difficult to design in advance an algorithm the makes the best choices with respect to any environment. On the other hand, there are known methods for adapting choices to an environment by learning over time with feedback. What seems to be missing is the capability of a designer to automatically have the program learn what would be the best to do.

2. Summary of Invention: Briefly describe the core idea of your invention (saving the details for questions #3 below). Describe the advantage(s) of using your invention instead of the known solutions described above.

The invention is a method by which a software engineer endows a software program with the ability to improve itself in the sense of making better choices with respect to the environment in which it works. The essence of the idea is to allow the designer just to indicate the possibilities for adaptation and let a learning algorithm find the best choices wherever the software designer thinks there is room for optimization. One way to implement this idea is to have a pre-compiler extract the learning instructions and generate a C program that includes all the learning components. The attached document contains a more detailed description of the method and a preferred embodiment.

autonomicprogramming.p

3. Description: Describe how your invention works, and how it could be implemented, using text, diagrams and flow charts as appropriate.

***Patent Value Tool**

* 1. Select the single most appropriate technology category for your invention from the following technologies list.
(610) PPM 600 Software/Services/ Applications/Solutions-610 Artificial Intelligence / Expert Systems
Comments
Are there any additional significant markets where the invention is likely to have impact?
[✓ Yes] No

Please identify them:
Autonomic computing

*2. Have you implemented the invention (e.g., made a prototype) or otherwise shown that it is workable?
[✓ Yes] No

*3. Has the subject matter of the invention or a product incorporating the invention been offered for sale.

or is it likely to be offered for sale, as part of an IBM product or service?
[✓ No known product plans within 2 years]
○ Maybe; GA 1-2 years away
○ Yes; GA within 3-12 months
○ Yes; GA within 3 months
○ Yes; product has been announced

*4. Has the invention been commercially used (internally or externally) by IBM or another entity (e.g., included in or used to make products, or prototypes provided to a customer)?
○ Yes [✓ No]

*5. In what type of product might a competitor include the invention?
Self-optimizing programs

What competitor(s) (indicate home country of such competitors if not United States)?
Software developers world wide

*6. How easily can the use of the invention by a third party be detected?
○ Undiscoverable; third party must admit use for IBM to know
○ Difficult; e.g.; with reverse engineering or examination of available code
○ With work; e.g.; using test cases; but not reverse engineering
○ Easily; by running & viewing product operation
[✓ Trivally; without purchase of product; e.g.; by reading product literature]

*7. Is the invention applicable to a standard?
○ Yes [✓ No]

*8. Have you, or any of the other inventors, submitted this invention disclosure or a similar invention disclosure previously?
[✓ Yes] ○ No

Please provide the disclosure number: ARC8-2002-0057

**9. Please list the invention disclosures (previously submitted or about to be submitted), products, patents, or publications that you and the other inventors feel are the most relevant to your invention (e.g., pertaining to the problem you are solving, including other solutions to the problem), be they from you or anyone else, or if not applicable, enter "None":
None other than ARC8-2002-0057

* 10. Was the invention made in the course of any activity that involved any other party, be it
- The government
- A customer (such as an RFQ)
- A development partner
- An alliance
- Any contract activity
- As part of a standards setting activity
- Other persons not employed by IBM

○ Yes [✓ No]

*11. Have you ever disclosed your invention to anyone outside IBM, or do you plan to do so in the future?
○ Yes [✓ No]

*12. If the invention relates to a product or service that is outside the scope of your business unit, please recommend IBM business unit(s), IBM location(s) or individual(s) within IBM that you think would provide a competent evaluation of your invention:

*PVT II

All of the questions below are required and must be answered in order to calculate a PVT Score
A.Threshold Questions

[redacted]

*1. **Operability** - Is there an identifiable operable embodiment of the invention (i.e., an embodiment that has been demonstrated or that would be reasonably expected to provide the benefits of the invention)?
[✓ Yes] ○ No

Reasons for above answer:

*2. **Novelty**- Are one or more concept(s) of the invention novel over what is already known in the literature, existing commercial products, patents, and earlier IBM invention disclosures?
[✓ Yes] ○ No
Reasons for above answer:

B.Valuation Questions

*1. Adequacy of Description:
○ Inadequate; invention unclear from description
○ Incomplete; essential features missing
○ Further clarification or implementation detail needed
[✓ Clear and complete as is]
State reason for answer:

*2. Technical contribution of invention:
○ None
○ Minor addition to known technology
[✓ Significant addition to known technology]
○ Major advance in technology
Reasons for above answer:

*3. Describe the problem solved/benefit provided and the implementation cost of the invention compared to existing or reasonably expected alternatives:
○ Minor problem/incremental benefit - significant implementation cost
[✓ Significant problem; substantial benefit - significant implementation cost]
○ Minor problem/incremental benefit - minor implementation cost
○ Significant problem/substantial benefit - minor implementation cost

*4. Are any alternatives to the invention available to those wishing to avoid its use?
○ Suitable alternatives available
[✓ Alternatives have drawbacks]
○ No feasible alternatives
Reasons for above answer:

*5. Describe the likelihood of use of the invention (answer each):

IBM's customers?	○ Unlikely	[✓ Possible]	○ Probable	○ Definite
IBM's suppliers/vendors?	○ Unlikely	[✓ Possible]	○ Probable	○ Definite
IBM's competitors?	○ Unlikely	[✓ Possible]	○ Probable	○ Definite
IBM?	○ Unlikely	[✓ Possible]	○ Probable	○ Definite

Reasons for above answer:

*6. What % of third party products in the technical field will likely contain the invention?
[✓ < 25%]
○ 25-50%
○ 50-75%
○ > 75%
Reasons for above answer:

IBM Confidential [redacted]

*7. How long is the invention likely to be used in products by IBM or others?
○ < 5 years
○ 5-10 years
○ 10-15 years
[✓ > 15 years]
Reasons for above answer:

*8. How easily can use of the invention by a third party be detected?
○ Undiscoverable; third party must admit use for IBM to know
[✓ Difficult; e.g.; with reverse engineering or examination of available code]
○ With work; e.g.; using test cases; but not reverse engineering
○ Easily; by running & viewing product operation
○ Trivially; without purchase of product; e.g.; by reading product literature
Reasons for the above answer, including description of how use could be detected:

Post Disclosure Text & Drawings

To add additional information related to this disclosure once it has been submitted, click the action button below and a new document will be opened for you to enter the new information. To view existing post disclosure information, double-click on the item in the list below (if there has been additional information entered), and the document will open for you to view.

Date entered	Post disclosure information (comments and drawings)
[redacted]	Comments by Nimrod Megiddo
[redacted]	Comments by Nimrod Megiddo

Form Revised 09/01/02)

A Method for Autonomic Optimization by Computer Programs

Nimrod Megiddo

[redacted]

1 Introduction

Software engineers often have various alternate ways of implementing algorithms and they fix choices within the code based on current beliefs about future environments. For example, an online scheduling algorithm picks tasks for execution, attempting to optimize the system according to some criterion, but the code designer may not know what the best choices are in varioues situations. Quite often, the code designer lacks the expertise that is required for making the good choices. In many situations, the good choices depend on the environment and it is quite difficult to design in advance an algorithm the makes the best choices with respect to any environment. On the other hand, there are known methods for adapting choices to an environment by learning over time with feedback.

Our purpose here is to formulate a framework in which a software engineer endows a software program with the ability to improve itself in the sense of making better choices with respect to the environment in which it works. The essence of the idea is to allow the designer just to indicate the possibilities for adaptation and let a learning algorithm find the best choices whereever the software designer thinks there is room for optimization. One way to implement this idea is to have a pre-compiler extract the learning instructions and generate a C program that includes all the learning components.

We first describe the pre-compiler commands and then explain how the learning is performed.

2 The pre-compiler commands

In this section we describe a set a posssible pre-compiler command which support the specification if the learning tasks.

2.1 The !CHOOSE command

The execution of computer is essentially deterministic, except that probabilistic computation can be implemented with pseudo-random number generators. We propose here an extension to programming languages where the engineer leaves some choices open so that the program can later make good choices with respect to the environment in which it runs.

A !CHOOSE command can be included in the program for to be handled by a pre-compiler as follows. It resembles the `switch` command in C, except that the processor is free to choose (at execution time) *any* one of the functions in the list between !CHOOSE command and !ENDCHOOSE, ignoring the label and the parenthesized list of variables. The variables within the parentheses are used to encode a "state" which is suggested to the processor as a base for making the choice[1]. The idea is that the program would be able to learn over time the good choices with respect to the observable quantities. For example,

```
!CHOOSE      label1
             (x1, x2, y1, z2)
             1 : f1();
             2 : f2();
             3 : f3();
!ENDCHOOSE
```

Any one choice of a function should result in a valid execution of the program, but may cause the program to perform differently and may yield different results. The compiler is instructed either to build a program that learns from its experience or rather make arbitrary choices (i.e., choose the first function in each case).

2.2 The !RULES command

Another pre-compiler feature is the capability of supplying specific rules of how to make a choice based on the state. This may be useful when a program has already gained some knowledge, and a small modification requires

[1] We discuss later the decision rules.

recompilation, so the knowledge can be passed to the revised program. One possible implementation of this feature is

```
!RULES   label1   func()
```

where `label1` is the label of the choice point and `func(x1,x2,x3)` is a user-supplied function of the state variables of that decision point, which selects one the available choices.

2.3 The `!REWARD` command

If a program is expected to optimize, it must be told what the goal is or at least be informed of rewards that are compatible with the goal. The reward mechanism may be implemented as follows. The command

```
!REWARD   label1   func1()
```

may be placed anywhere in the program. The call specifies which point of choice (e.g., label1) is rewarded, and which user-supplied function (func1) is to be used for calculating the reward.

3 Generating optimizing program

If the pre-compiler is instructed to generate an optimizing program, then it generates a C program that incorporates learning algorithms (e.g., Q-learning, TD-learning) for developing a good policy of making choices based on the current values of the state variables at each choice point.

One possible way of learning is as follows. Suppose the state variables at a certain choice points are $s_1, \ldots, s_n$ and suppose a choice of function has to be made from the indices $1, \ldots, m$. The state variables may themselves be functions of other program variables, as the desginer may wish. Denote the "value" a choice i ($i = 1, \ldots, m$) at the state $s = (s_1, \ldots, s_n)$

$$V(i, s) = \sum_{j=1}^{n} \alpha_{ij} s_j$$

where the α_{ij}'s are supposed to be learned by the system. The linearity assumption is not too restrictive since the state variables themselves may be functions of other variables. The rewards associated with choice point are used to update the current estimates of the coefficients α_{ij} that determine the $V(i, s)$s. The best choice at s is an i which maximizes $V(i, s)$. However, during the learning phase the system explores different values of i and uses known learning algorithms for approximating the true values of the $V(i, s)$s. The current α_{ij}s may be used for detremining the actual choice of the algorithm (with respect to the state) once the learning phase has ended, and the compiler generates a deterministic program.